Bounds and Improvements for BiBa Signature Schemes

Bounds and Improvements for BiBa Signature Schemes

Michael Mitzenmacher
and
Adrian Perrig

TR-02-02

Computer Science Group
Harvard University
Cambridge, Massachusetts

Bounds and Improvements for BiBa Signature Schemes

Michael Mitzenmacher and Adrian Perrig

Abstract

This paper analyzes and improves the recently proposed bins and balls signature (BiBa [23]), a new approach for designing signatures from one-way functions without trapdoors.

We first construct a general framework for signature schemes based on the balls and bins paradigm and propose several new related signature algorithms. The framework also allows us to obtain upper bounds on the security of such signatures. Several of our signature algorithms approach the upper bound. We then show that by changing the framework in a novel manner we can boost the efficiency and security of our signature schemes. We call the resulting mechanism *Powerball* signatures. Powerball signatures offer greater security and efficiency than previous signature schemes based on one-way functions without trapdoors.

Keywords: One-time signature, signature based on one-way function without trapdoor, Powerball signature.

1 Introduction

Although the speed of high-end processors continues to steadily increase, we simultaneously witness the proliferation of low-powered, resource-starved handheld devices (e.g. cell phone, pager, Palm pilot). These handheld devices are designed for mobility and convenience, and their computation power is limited by minimal microprocessors and energy resources.[1] Similarly, low powered computation devices have been proposed to build sensor networks for measuring the weather or other geographically distributed phenomena. We collectively call handheld devices and sensor network nodes with constrained computation and energy resources *small devices*. The widespread deployment of small devices with severe resource constraints motivates the need for faster and simpler signature mechanisms, even though microprocessors continue to dramatically increase in speed. On these small devices, even the most efficient asymmetric signature algorithms typically require on the order of seconds to generate or verify a signature (assuming that the signature code even fits into memory). Section 7 reviews related work in efficient signature schemes.

Signatures based on one-way functions without trapdoors (sometimes called *one-time signature schemes*) are an interesting alternative to signatures based on asymmetric cryptography [4, 5, 14, 18, 19, 28]. One of their main advantages is that these signatures only rely on a one-way function, which we can implement with a fast hash function (e.g. SHA-1 [22] or MD5 [29]), or from a block cipher [16, 20, 26, 27].

These one-time signature schemes are orders of magnitude faster than traditional signatures, so they may be an attractive alternative for small devices. However, some of these schemes have large

[1] To save production costs, manufacturer deploy minimal microprocessors for the required task. Even in the year 2000, 80% of all microprocessors shipped are 4-bit and 8-bit processors [35].

signatures, and can only sign a fixed number of messages per public key. We review the merits and drawbacks of one-time signature schemes in Section 7.

The recently proposed bins and balls (BiBa) signature is a promising new approach to mitigate some of the drawbacks of one-time signatures [23]. We review BiBa in Section 2. In Section 3 we present an abstract framework for these types of signatures, which allows us to present new approaches in Section 4 that are more secure. Our framework also allows us to analyze the security of these signature schemes (see Section 5); we find that the security of the basic BiBa signature as well as several of our variations is close to the theoretical bound.

In Section 6 we extend our abstract framework and find an opportunity for a new signature scheme that improves the security of the previous approach (given a certain signature overhead). From this framework, we derive the *Powerball* signature, a new one-time signature scheme with low overhead and high security. We find that Powerball schemes are viable alternatives for signatures in small devices.

2 Review of the BiBa Signature

This section presents a brief review of the BiBa (bins and balls) signature algorithm [23]. The set of t secret balls constitutes the private key $\mathcal{PK}^{-1} = \{B_1 \dots B_t\}$. The public key commits to all balls in the private key.[2] The public key may be the concatenation of t commitments $\mathcal{PK} = F(B_1) \mid\mid \dots \mid\mid F(B_t) = c_1 \mid\mid \dots \mid\mid c_t$, or the public key may be the root of a Merkle hash tree computed over the secret balls [17]. For simplicity of the following description, we assume that the public key is the concatenation of commitments.

To sign message M, the signer computes the hash of the message $h = H(M)$ and uses h to select a one-way function g_h from a family of hash functions G (in the random oracle model [2]). The hash function g_h maps each ball to one of the n bins. The signature is a collection of balls that produce a special pattern in the bins. A BiBa signature is a collection of k balls that form a k-way collision under g_h in one bin: $\langle B_{\alpha_1}, \dots, B_{\alpha_k} \rangle$ where α_i is the index in the public key of the ith ball in the signature.

To verify the signature $\langle B_{\alpha_1}, \dots, B_{\alpha_k} \rangle$ on message M, the verifier performs the steps: (1) check that all balls of the signature are distinct ($B_{\alpha_i} \neq B_{\alpha_j}$ for $i \neq j$); (2) verify the authenticity of the balls using the public key (check that $F(B_{\alpha_i}) = c_{\alpha_i}$[3]); (3) compute $h = H(M)$ and select g_h from the one-way function family; (4) verify the k-way collision ($g_h(B_{\alpha_1}) = \dots = g_h(B_{\alpha_k})$).

Note that the probability P_s that the signer can successfully sign a message is less than 1. To deal with this problem, the signer can use a counter value c as follows. The signer computes the hash of the message $h = H(M \mid\mid c)$ and uses h to select the one-way function g_h. If this does not lead to a successful signature the signer can increment the counter and try again. The signature is then $\langle B_{\alpha_1}, \dots, B_{\alpha_k}, c \rangle$. In this setting we may define P_s to be the probability that the signer can successfully sign for a given value of c. In the original BiBa paper a design goal was to have $P_s \geq 1/2$, so that on average only two values of c need to be tried.

[2] A commitment locks in a secret s without revealing s. We use a one-way and weak collision resistant function F to commit to a secret s: the commitment is $c = F(s)$. To open the commitment, one publishes s and anybody can verify that s really corresponds to c: compute $F(s)$ and verify the equality $c = F(s)$.

[3] In practice the singer could help the verifier by also sending the indices α_i; this does not change the security of the system since the the forger could easily change these uncommitted values.

k	n	P_f	k	n	P_f
2	762460	$2^{-19.5403}$	13	192	$2^{-91.0196}$
3	15616	$2^{-27.8615}$	14	168	$2^{-96.1001}$
4	3742	$2^{-35.6088}$	15	151	$2^{-101.3377}$
5	1690	$2^{-42.8912}$	16	136	$2^{-106.3119}$
6	994	$2^{-49.7855}$	17	123	$2^{-111.0802}$
7	672	$2^{-56.3539}$	18	112	$2^{-115.7250}$
8	494	$2^{-62.6386}$	19	104	$2^{-120.6079}$
9	384	$2^{-68.6797}$	20	96	$2^{-125.1143}$
10	310	$2^{-74.4851}$	21	89	$2^{-129.5147}$
11	260	$2^{-80.2237}$	22	83	$2^{-133.8758}$
12	222	$2^{-85.7386}$	23	78	$2^{-138.2788}$

Table 1: The security of some BiBa instances. The signer knows $t = 1024$ balls and the adversary has $r = k$ balls. The table shows the probability of forgery P_f to find a k-way collision when throwing k balls into n bins.

We define P_f as the probability that an attacker forges a signature after one attempt. We list Table 1 from [23], which lists P_f for different BiBa instances, where in each case the number of bins n is chosen so $P_s \approx 1/2$. To compute the P_f listed in the table, we assume that the attacker knows the balls from one disclosed signature, so $P_f = \frac{1}{n^{k-1}}$.

3 A generalized setting

We may abstract the BiBa setting into a combinatorial balls and bins setting as follows. The signer has t balls, $B_1, B_2, \ldots, B_t$, from a universe U_1. The signer can construct functions g_{h_i}, for $i = 1, 2, \ldots$, so that the functions g_{h_i} map balls into bins, where the bins lie in a universe U_2. We assume that the functions g_{h_i} look random, so that we model the bin $g_{h_i}(B_j)$ as a bin chosen independently and uniformly at random from U_2. A signature consists of a function index i along with a set of k ordered pairs of balls and their corresponding bins, $\{(B_{\alpha_1}, g_{h_i}(B_{\alpha_1})), (B_{\alpha_2}, g_{h_i}(B_{\alpha_2})), \ldots, (B_{\alpha_k}, g_{h_i}(B_{\alpha_k}))\}$. For a signature to be valid, it must be a member of the set of valid signature patterns P, where $P \subseteq (U_1 \times U_2)^k$.

A forger can construct functions g_{f_i}, for $i = 1, 2, \ldots$, that also appear to map balls to bins independently and uniformly at random. The forger, however, does not have access to all t balls, but only to balls used by the sender in a sent signature. In the case where a set of t balls is only used to construct a single valid signature, the forger will only have access to k balls; this is the case we consider in detail here. The forger creates a successful signature forgery $\{(B_{\alpha_1}, g_{f_i}(B_{\alpha_1})), (B_{\alpha_2}, g_{f_i}(B_{\alpha_2})), \ldots, (B_{\alpha_k}, g_{f_i}(B_{\alpha_k}))\}$ if this signature lies in the set of valid signature patterns P.

In the original BiBa paper, the set of valid signature patterns P consisted of any set of k balls that all fell in the same bin. The design goal that the successful signature probability P_s be at least $1/2$ determines the number of bins n that can be used. While this choice is somewhat arbitrary, as we shall see in our analysis in Section 5, it is useful for comparison purposes and we will adopt it hereon as well.

4 Variations

In this section we consider other possible schemes based on our general framework. Some of our examples prove better than the original BiBa scheme in some cases; others are given simply as instructional examples of what is possible. The variations listed are by no means exhaustive.

We provide limited numerical results, comparing our scheme against the BiBa scheme for the range of P_f values that are interesting for practical applications (roughly 2^{-70} to 2^{-90}).

- The bins correspond to the range $[0, n-1]$ and a valid signature pattern consists of k balls that lie in distinct bins $a_1, a_2, \ldots a_j$ with k_i balls in bin a_i. As a specific example, the k balls could lie in two distinct bins each with exactly $k/2$ balls. This is a natural generalization of the BiBa scheme that performs better for some parameter settings.

  For example, let us consider the case where a valid signature pattern consists of k balls with $k/2$ balls in each of two distinct bins. The probability the forger succeeds is $\binom{n}{2}\binom{k}{k/2}\left(\frac{1}{n}\right)^k$; this is easily seen by multiplying the number of ways of choosing two bins, the number of ways of splitting the k balls between the bins, and the probability the balls land in the appropriate bins.

  In Table 2 below, we consider some specific examples where at least x balls are required to land in each of y distinct bins, so that $k = x \cdot y$. In all of these simulation results, we check that $P_s \geq 1/2$ over a series of $1,000,000$ trials.[4] In all cases using two bins with $k/2$ balls a signature performs better than the original BiBa scheme by at least a factor of 2.

k	x	y	n	P_f
10	5	2	1308	$2^{-75.85}$
12	6	2	796	$2^{-87.52}$
14	7	2	551	$2^{-98.53}$
12	4	3	2290	$2^{-87.96}$
12	3	4	6856	$2^{-87.57}$

Table 2: Results when a signature requires throwing k balls into y bins with x balls in each bin.

- The bins correspond to the range $[0, n-1]$ and a valid signature consists of k balls falling in sequential bins modulo n. For the forger, the probability that k balls form a signature is just $(1/n)^{k-1}k!$; there are n possible starting positions, and for each starting position the probability the k balls land in the appropriate k consecutive bins in some order is $(1/n)^k k!$. This scheme appears to perform slightly worse than the original BiBa scheme, as shown in Table 3, which is also based on having $P_s \geq 1/2$ over $1,000,000$ trials.

- The balls lie in a universe $[0, 2^z)$, the bins correspond to the range $[0, n-1]$, and a valid signature pattern consists of k balls $B_{\alpha_1} < B_{\alpha_2} < \ldots < B_{\alpha_k}$ falling in sequential bins in order. That is, B_{α_1} falls in the first bin in the sequence, B_{α_2} in the second, etc. This extends

[4] Technically, ensuring $P_s \geq 1/2$ requires some statistical care; in practice, we simply tested that if $P_s \leq 1/2$, we were obtaining results at least one standard deviation from the mean. Small variations in n yield minor variations in P_f, so we feel our results are accurate enough for demonstrative purposes.

k	n	P_f
10	1489	$2^{-73.07}$
11	1318	$2^{-78.39}$
12	1188	$2^{-83.52}$
13	1087	$2^{-88.50}$

Table 3: Results when a signature requires k consecutive non-empty bins.

k	n	P_f
10	290	$2^{-73.62}$
11	241	$2^{-79.13}$
12	205	$2^{-84.47}$
13	177	$2^{-89.61}$

Table 4: Results when a signature requires k consecutive bins with balls in temporal order.

the previous example to include a natural temporal ordering on the balls. One might think the signer would have an advantage in this case since the sender can have several balls in a bin, and therefore the effect of the temporal ordering may be harsher for the forger than the signer. Note the probability of a forgery is now just $(1/n)^{k-1}$, matching the original scheme.

This modification improves over the previous scheme slightly but the resulting numbers are still not better than BiBa, as shown in Table 4. Again, the results are based on $1,000,000$ trials.

- The n bins correspond to $\binom{v}{2}$ edges on a graph with v vertices, and a valid signature pattern consists of k edges that form a cycle. While this scheme sounds simple, in practice it would prove hard to implement. While algorithms for finding k-cycles in graphs exist, the best known general algorithms are exponential in k [1, 39]. (Since these are random and fairly sparse graphs, better algorithms may exist; still, this is a non-trivial problem.) Since cycles of length 4 are easier to find, we suggest the following variation.

- The n bins correspond to $\binom{v}{2}$ edges on a graph with v vertices, and a valid signature pattern consists of $k = 4k_1$ edges that form k_1 vertex-disjoint cycles of length 4. Finding cycles of length four can be done using matrix multiplication on the adjacency graph, and faster algorithms are known [10]. This approach still requires significant computation for finding a signature, unlike the original BiBa scheme; however, verifying a signature can still be done quickly.

  We consider the specific case of $k = 12$ and compute the probability of a successful forgery. There are $\frac{1}{6}\binom{v}{4}\binom{v-4}{4}\binom{v-8}{4}$ possible ways of choosing the sets of vertices that constitute the three cycles, and then three ways of orienting the vertices within a cycle. Hence the probability of a successful forgery is

$$\frac{\frac{27}{6}\binom{v}{4}\binom{v-4}{4}\binom{v-8}{4}12!}{\binom{v}{2}^{12}}$$

In simulations we find that 936 vertices yields $P_s \geq 1/2$. We did only simulations of 10,000 trials here, as we used simple matrix multiplication techniques to check for cycles of length four. In this case $P_f = 2^{-89.28}$. This is more than a factor of eight smaller than for the original BiBa scheme.

- The balls lie in a universe $[0, 2^z)$, and the bins correspond to the range $[0, n-1]$ for an even number n. We assume the balls are thrown in sequential order, according to a load balancing scheme as described by Vöcking [37]. Each ball has two possible hash locations, one chosen independently and uniformly at random from the range $[0, n/2-1]$ (which we call the left) and the other chosen independently and uniformly at random from the range $[n/2, n-1]$ (which we call the right); it is placed in the bin with fewer balls, with ties being broken in favor of the smaller numbered bin (toward the left). A signature in this scheme corresponds to a *witness tree*, which proves that a bin with a certain number of balls exists. For example, to show that a bin on the left holds three balls, we must not only show the three balls in that bin, but we must show for the third ball on the left that the corresponding bin on the right had two balls there previously. Further discussion of the witness trees can be found in [37], and of course this approach can be generalized to other similar hashing schemes.

  The specific case of $k = 12$ corresponds to a witness tree for a bin with three balls on the left, where there are no repeated balls in the tree. We tested this case, finding that 1316 bins allow for $P_s \geq 1/2$. The probability of a false positive is somewhat more difficult to compute; we simply note that with these parameters $P_f = 2^{-87.68}$, which is almost a factor of 4 better than the corresponding BiBa scheme.

5 A Unifying Analysis

It should be apparent from our results in the previous section that many of the schemes we suggest appear to perform nearly the same. This may seem somewhat unusual, given the variety in the descriptions of the schemes and the variety in the number of balls necessary to achieve $P_s \geq 1/2$. In this section we provide an analysis that explains this behavior. Our analysis yields both an upper bound on and an approximation for the performance of BiBa schemes and the variations we have considered in Section 4.

We will say that a bin is *covered* if a ball lands in the bin. Let us first consider any balls and bins setting where each successful signature corresponds to one of N distinct patterns, where each pattern consists of a list of k distinct bins to be covered.

Theorem 1 *In the setting where a valid signature corresponds to one of N distinct patterns, where each pattern consists of a list of k distinct bins to be covered,*

$$\frac{P_s}{P_f} \leq \frac{t^k}{k!}$$

Proof: We first note that the probability of success for the forger is $P_f = \frac{Nk!}{n^k}$. Now consider the probability of success for the signer. As an upper bound (and rough estimate) for the success of the signer, we may consider the expected number of successful patterns matched by the signer. To

see this, let p_i be the probability that the signer matches *at least* i patterns, and let X be a random variable representing the number of patterns matched. Then

$$E[X] = p_1 + p_2 + p_3 + \ldots$$

Hence $E[X] \geq p_1$ (and in fact $E[X] \approx p_1$ when p_i is small for $i \geq 2$).

Now consider the event that for a specific pattern all k bins are covered. The probability that any single bin is covered is at most t/n by a union bound. Moreover, for any two bins, the events corresponding to each being covered are negatively correlated. It follows easily that $(t/n)^k$ is an upper bound on the probability that all bins in the pattern are covered. Hence $N(t/n)^k \geq E[X] \geq p_1 = P_s$.

It follows that $\frac{P_s}{P_f} \leq \frac{t^k}{k!}$, proving the theorem. ■

Interestingly, this upper bound is independent of the number of bins n and the number of patterns N.

Looking at the argument more closely, we see that the upper bound should be a fairly good approximation of the ratio. There is an error introduced because $E[X] \geq p_1$, but when p_i is small for $i \geq 2$, this error is not large. Also, in bounding $E[X]$ we used a union bound of t/n for the probability that a bin is covered. In fact the probability that any specific bin remains uncovered $(1 - 1/n)^t \approx \mathrm{e}^{-t/n}$. Now if n is large, the events corresponding to bins being covered are nearly independent. Hence for sufficiently large n, the probability that k bins that constitute a pattern are covered is approximately $(1 - \mathrm{e}^{-t/n})^k$. If n is large compared to t, then this is approximately $(t/n)^k$, the quantity used in the theorem.

Hence we conclude this upper bound is a good approximation when n is large compared to t and when p_i is small for $i \geq 2$. These properties are approximately true for many of our variations, explaining their similar performance despite the varying nature of the patterns and the number of bins required to achieve a success probability $P_s \geq 1/2$. This argument also explains why the variations that have more bins generally appear to do better than the original BiBa scheme. The poorer performance of schemes involving covering several consecutive bins is also clarified, as with these schemes it is clear that p_2 and higher values of p_i are comparatively large.

While technically the above argument assumed that a pattern consisted solely of a set of bins to be covered, entirely similar results can be shown to hold for all of the variations we have considered. For example, consider the original BiBa scheme, in which a bin is supposed to receive not just one but many balls, which does not appear to fit this model. However, consider the relationship between an original BiBa scheme with n bins and a modified scheme with ng bins that are grouped into n groups of size g. If we seek k balls in the same bin for the original BiBa scheme, then our patterns in the modified scheme will consist of all sets of k distinct bins that all lie in the same group. The two schemes are nearly equivalent, and hence the performance ratio is essentially the same.

Similarly, requiring the balls to arrive in a specific order does not change the result. The probability of success for the forger drops to to $\frac{N}{n^k}$, since ordering variations no longer help the forger. But there is a corresponding drop in the bound for P_s by a $1/k!$ factor, since the sender must also achieve a specific ordering on the balls.

6 The Powerball Signature

This section introduces the Powerball signature, our improvement on the BiBa signature. Our new signature is based on the following observation. The original BiBa scheme has a fixed number of known signature patterns, i.e., a collision of k balls in one bin is a valid signature pattern. In BiBa, these patterns are implicit; all the participants agree on them. In our new approach, the signature patterns are explicit. In the same way the signer commits to t balls in the public key, the signer also commits to t' patterns P_i ($1 \leq i \leq t'$). Each pattern specifies k bins, so $P_i = \langle b_1, \ldots, b_k \rangle$.

As in BiBa, to sign message M, the signer computes the hash of the message $h = H(M \mid\mid c)$ (c is a counter that the sender increments if it didn't find a signature) and uses h to select a one-way function g_h from a family of hash functions G (in the random oracle model [2]). The hash function g_h maps each ball to one of the n bins. To find a valid signature, the signer searches for a *complete pattern* P_i, where every bin in the pattern contains a ball. (If a bin appears β times in the pattern, the corresponding bin contains at least β balls.) If the signer finds a complete pattern P_i, it creates the signature $\langle B_{\alpha_1}, \ldots, B_{\alpha_k}, P_i, c \rangle$ (where α_j are the indices of the balls that landed in the bins of pattern P_i).

To verify the signature $\langle B_{\alpha_1}, \ldots, B_{\alpha_k}, P_i, c \rangle$ on message M, the verifier performs the steps: (1) check that all balls of the signature are distinct ($B_{\alpha_i} \neq B_{\alpha_j}$ for $i \neq j$); (2) verify the authenticity of the balls using the public key (check that the commitment $F(B_{\alpha_i})$ is in the public key); (3) verify the authenticity of the pattern P_i using the public key (check that the commitment $F(P_i)$ is in the public key); (4) compute $h = H(M \mid\mid c)$ and select g_h from the one-way function family; (5) verify that the k balls cover all k bins of pattern $P_i = \langle b_1, \ldots, b_k \rangle$, so $g_h(B_{\alpha_1}) = b_1, \ldots, g_h(B_{\alpha_k}) = b_k$.

Let us consider the ratio of success between the sender and the forger in this model. The forger knows k balls and a pattern. Recall that in the standard scheme with $k+1$ balls sent, we found an upper bound on this ratio $\frac{t^{k+1}}{(k+1)!}$. The probability of success for the forger in our new scheme is $P_f = \frac{k!}{n^k}$.

For the signer, we approximate the expected number of matched patterns, which in turn approximates P_s. For simplicity we assume that the signer has $t' = t$ possible patterns; we further assume that the system is arranged so that these patterns are distinct. As before, the probability that each is covered is upper bounded by $(t/n)^k$; this is a good approximation if n is much larger than t. Hence our approximation for P_s is t^{k+1}/n^k, and hence the ratio between the sender and forger is $\frac{t^{k+1}}{k!}$. Our new scheme therefore changes the bound of Theorem 1, doing better by a factor of $k+1$.

Adding $t' = t$ commitments of the patterns to the public key would double its size, a rather severe additional cost. We introduce a method to add the patterns to the public key without increasing its size. Imagine that the ball is the commitment of the pattern, so a commitment in the public key commits to both the ball and the pattern. We call this structure a *Powerball*. For a Powerball, we begin with a bit string that represents a pattern P_i. (For now we assume a simple mapping from bit strings to patterns.) The ball B_i is derived from the pattern P_i using the one-way function F: $B_i = F(P_i)$. The commitment C_i is then a further application of F on B_i: $C_i = F(B_i) = F(F(P_i))$. This requires the additional assumption that F is not only one-way, but that as a function it appears random, so that we may assume the balls are distributed independently and uniformly at random. Hash functions in the random oracle model have this property [2].

Note that the forger can obtain a $(k+1)$st ball from P_i by computing $B_i = F(P_i)$. We solve

k	n	P_f
9	1734	$2^{-78.37}$
10	1548	$2^{-84.17}$
11	1407	$2^{-89.79}$
12	1295	$2^{-95.23}$
13	1204	$2^{-100.50}$

Table 5: Results with the Powerball scheme when a signature pattern uses k bins, and therefore $k+1$ Powerball are used.

this problem by requiring that the ball B_i does not occur as a ball in the signature. If the forger does not have another pattern, it cannot use B_i because it has to use the only pattern it knows.

Results from simulations of the Powerball scheme are presented in Table 6. Comparing with Table 1, we see that the Powerball scheme does improve performance, as the theoretical framework suggests. A Powerball is worth almost another ball; that is, using $k+1 = 11$ Powerballs is almost as good as requiring 12 balls to fall into a bin using the original BiBa scheme.

We can slightly enhance the advantage for the signer by further changing the meaning of a Powerball. For example, suppose we require that two Powerballs must be combined in some order to represent a pattern. For example, we may take the exclusive-or of bits in the P_i in order to obtain a pattern. In this case we use $k+2$ Powerballs to represent a signature; k correspond to balls, and two correspond to a pattern. In this case we still have $P_f = \frac{k!}{n^k}$. On the other hand, for the signer we have $E[X] \approx \frac{t^{k+2}}{2n^k}$. Note the introduction of the factor of two in the denominator, since there are $\binom{t}{2}$ possible patterns for the signer. Hence the upper bound on the ratio P_s/P_f is about $t^{k+2}/2k!$. This is a factor $\binom{k}{2}$ better than the scheme without Powerballs. Again, there are tradeoffs to using such mechanisms, including the difficulty for the signer to find a matched pattern, so these Powerball variations may be of theoretical interest only. However, this demonstrates how small changes in the model can lead to different analyses.

A similar idea can be used to reduce the size of the public key, which is very large in the standard BiBa scheme. Suppose we require that two Powerballs be combined, say via an exclusive-or, in order to construct a ball. In this case, a sender with t Powerballs has roughly $\binom{t}{2}$ balls to play with; this number is not exact because we restrict each pair of Powerballs to be disjoint. Now a forger with k non-pattern Powerballs has $(k-1)\cdot(k-3)\cdot\ldots 3\cdot 1$ ways of pairing up the k Powerballs into $k/2$ actual balls. Hence at the cost of increasing the power of the forger somewhat (by giving the forger more than one set of balls to use), we can dramatically reduce the size of the public key. Whether this tradeoff is useful may depend on the desired system parameters.

7 Related Work

We first review related work in efficient asymmetric signatures targeted towards resource-constrained devices. We then review research related to signatures based on one-way functions without trapdoors. We also point out that the idea of using the asymmetry between signers and forgers in balls and bins scenarios has been used in other situations, such as the MicroMint payment scheme [30].

Efficient Signature Algorithms for Resource-constrained Devices

Previous studies show that computing asymmetric cryptographic operations (e.g. computing an RSA signature [31], or a Diffie-Hellman key agreement [9]) takes on the order of seconds and sometimes even minutes on some handheld devices. Brown et al. analyze the computation time of various digital signature algorithms on various platforms [7]: Elliptic Curve Cryptography (ECC) signature algorithms require 1.0–2.2 seconds for one signature generation, and 1.8–5.3 seconds for verification (on a Palm Pilot or RIM pager). On the same architecture, a 512-bit RSA signature requires 2.4–5.7 seconds for generation, and 0.1–0.6 seconds for verification (depending on the public exponent). The problem of performing cryptographic operations on minimal hardware is even more pronounced on some sensor networks. For example, futuristic *Smart Dust* sensors present even more stringent resource constraints [13, 38].

To speed up the slow signature generation, Even, Goldreich, and Micali propose on-line / off-line signatures [11]. The slow signing operation is performed off-line, and the signer has subsequently a low overhead to generate the final signature. They propose to use a traditional signature algorithm to sign the public key of a one-time signature algorithm off-line. The on-line signature with the one-time signature algorithm is very efficient.

Schnorr proposes a signature algorithm that allows the signer to perform most of the work off-line and the remaining on-line work is efficient [33]. Shamir and Tauman propose a signature based on chameleon hashing which allows off-line precomputation and efficient on-line signing [34].

Other researchers propose to use computationally more powerful third parties to off-load some of the expensive operations. For example, Modadugu, Boneh, and Kim propose to use an untrusted third party to speed up RSA key generation on a small device [21].

Smart cards also attracted attention for efficient signature algorithms. Poupard and Stern design signature algorithms efficient on smart cards [24, 25]. Courtois, Goubin, and Patarin also design new signature algorithms efficient for smart cards [8]. Lenstra and Verheul propose an efficient signature based on XTR, which provides short signatures [15]. Hoffstein, Pipher, and Silverman propose NSS, an efficient NTRU lattice-based signature algorithm [12]. To the best of our knowledge, the signature verification times of all of these algorithms are still slower than RSA.

Signatures based on One-way Functions without Trapdoors

Signatures based on one-way functions without trapdoors are sometimes also called one-time signatures.

Rabin published the first one-time signature based on a symmetric encryption function [28]. The signature requires interaction between the signer and the verifier, and the public key and signature are on the order of 1 Kbyte.

Lamport shows how to construct a digital signature out of a one-way function [14]. His approach does not require interaction between the signer and verifier, however, the size of the validation parameters and signature are still on the order of 1 Kbyte. Lamport's basic approach is that the signer publishes two commitments for each bit (for 0 and 1, respectively) of the data to sign. To sign the message, the signer reveals one of the values previously committed to, based on whether the corresponding message bit was 0 or 1.

Merkle and Winternitz improved on Lamport's signature [18, 19]. Even, Goldreich, and Micali [11] use the Merkle-Winternitz approach to construct their on-line / off-line signature. Rohatgi

	Signature Generation		Verification	Signature size	Public key size
	Off-line	On-line	(expected)		
Lamport	160	1	80	80	160
Merkle-Winternitz	355	1	169	23	1
Bleichenbacher-Maurer	182	1	72	45	1
BiBa	1024	2048	23	11	1024
Powerball	2048	2048	20	10	1024

Table 6: Comparison of one-time signature algorithms. The table considers a signature of an 80-bit hash. For the Merkle-Winternitz signature, we use the parameters that Rohatgi proposes to sign 80 bits [32].

further refines Merkle and Winternitz's approach and proposes k-times signatures [32], which feature a small public key and allow signing k messages. The main drawback of this approach is the large signature size, which is around 300 bytes per signature (for a 6-times key), which is more than twice the size of the equivalent BiBa signature. Furthermore, the signer computes 350 off-line one-way function applications per signature, and the verifier computes 184 one-way functions on average to verify the signature.

Bleichenbacher and Maurer analyzed signature algorithms with a minimal number of nodes in the graph [5, 4, 3].

Table 6 compares the various one-time signature algorithms. We consider the computation and communication overhead as a basis for comparison. We choose the signature parameters such that a forger has a probability of 2^{-80} to find a valid signature after one try. For the computation overhead, we consider the number of one-way function computations the signer needs to perform to compute the public key (off-line), and the expected number of one-way function computations the signer performs to actually generate the signature (on-line). For the verification overhead we list the expected number of one-way function computations the verifier performs to check the signature. For the computation overhead, we consider the size of the public key, and the size of a signature. We express the signature and public key size in number of nodes. In practice, each node may be on the order of 96–128 bits long.

8 Discussion and Conclusion

To the best of our knowledge, the Powerball signature is the fastest signature for verification. To achieve a probability of forgery of $P_f \approx 2^{-80}$, the verifier only needs to compute 20 one-way functions. This verification cost decreases further if we increase the number of balls of the signer. In the most extreme case, the signature only contains a single ball, and the verifier only computes two hash functions to verify the signature.

The Powerball signature also features shorter signatures than previous one-time signature algorithms. These improvements come at the cost of a larger public key and a higher signature generation overhead. However, the signature generation in Powerball is highly parallelizable — with enough processors Powerball only requires two sequential hash function computations.

Other features of Powerball include the small code size (as we can implement it based on

a block cipher), that the security does not rely on any unproven number-theoretic assumptions (the signature remains secure even if a fast factoring algorithm is invented), and the fact that the signature algorithm cannot be used as an encryption algorithm (advantage for certain export restrictions).

The Powerball signature has many applications. For example, the fast signature generation (with parallel processors) and super-fast verification may be useful in stock trading systems that require non-repudiation and the lowest possible end-to-end delay.

Another application is in small devices that take seconds to generate or verify a traditional asymmetric digital signature. Some embedded 8 bit microprocessors even lack a built-in multiplication instruction. Thus, many traditional signature algorithms are inefficient on such devices. Fortunately, many efficient block ciphers exist for these architectures, and we can implement Powerball based on a single block cipher encryption function.

The Powerball signature may also solve another hard problem. Many applications that rely on digital signatures are susceptible to a denial-of-service (DoS) attack: an attacker floods the victim with a large number of bogus signatures. Because signature verification is generally a slow operation (a 1024-bit RSA verify takes on the order of 0.5 millisecond on a 800 MHz Pentium II processor), the victim is computationally overwhelmed just checking all signatures. Powerball has a nice property: even if a forger can find a signature where $k-1$ balls land in the correct bin, a verifier that checks the balls of the signature in random order discovers the bad ball after checking after checking an average of $(k+1)/2$ balls. In practice, the forger can find even fewer matching balls, so the verifier can detect an invalid signature after a few hash function computations. The Powerball scheme is thus ideal to defend against these DoS attacks.

References

[1] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. In *Proceedings of the 2nd European Symposium on Algorithms*, number 855, pages 354–364, Utrecht, The Netherlands, 1994. Springer-Verlag, Berlin Germany.

[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *1st ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, Nov. 1993. ACM Press. Appeared also (in identical form) as IBM RC 19619 (87000) 6/22/94.

[3] D. Bleichenbacher and U. Maurer. On the efficiency of one-time digital signatures. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology – ASIACRYPT '96*, number 1163 in Lecture Notes in Computer Science, pages 196–209. Springer-Verlag, Berlin Germany, 1996.

[4] D. Bleichenbacher and U. Maurer. Optimal tree-based one-time digital signature schemes. In C. Puech and R. Reischuk, editors, *13th Symposium on Theoretical Aspects of Computer Science (STACS'96)*, number 1046 in Lecture Notes in Computer Science, pages 363–374. Springer-Verlag, Berlin Germany, 1996.

[5] D. Bleichenbacher and U. M. Maurer. Directed acyclic graphs, one-way functions and digital signatures. In Y. G. Desmedt, editor, *Advances in Cryptology – CRYPTO '94*, volume 839

of *Lecture Notes in Computer Science*, pages 75–82. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1994.

[6] G. Brassard, editor. *Advances in Cryptology – CRYPTO '89*, number 435 in Lecture Notes in Computer Science, Santa Barbara, CA, USA, 1990. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany.

[7] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M. Kirkup, and A. Menezes. PGP in constrained wireless devices. In *9th USENIX Security Symposium*, Denver, Colorado, Aug. 2000. USENIX.

[8] N. Courtois, L. Goubin, and J. Patarin. Flash, a fast multivariate signature algorithm. In D. Naccache, editor, *Progress in Cryptology - CT-RSA 2001*, number 2020 in LNCS. Springer-Verlag, Berlin Germany, Apr. 2001.

[9] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, Nov. 1976.

[10] C. Dorgerloh and J. Wirtgen. Once again: Finding simple cycles in graphs. Technical Report 85165-CS, University of Bonn, Germany, 1997.

[11] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In Brassard [6], pages 263–277.

[12] J. Hoffstein, J. Pipher, and J. H. Silverman. NSS: An NTRU lattice-based signature scheme. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT '2001*, number 2045 in Lecture Notes in Computer Science, pages 211–228, Innsbruck, Austria, 2001. Springer-Verlag, Berlin Germany.

[13] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for smart dust. In *International Conference on Mobile Computing and Networking (MOBICOM '99)*, pages 271–278, Aug. 1999.

[14] L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979.

[15] A. K. Lenstra and E. R. Verheul. Key improvements to XTR. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT '2000*, number 1976 in Lecture Notes in Computer Science, pages 220–233, Kyoto, Japan, 2000. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany.

[16] S. M. Matyas, C. H. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27:5658–5659, 1985.

[17] R. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, Apr. 1980. IEEE Computer Society Press.

[18] R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, number 293 in Lecture Notes in Computer Science, pages 369–378, Santa Barbara, CA, USA, 1988. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany.

[19] R. C. Merkle. A certified digital signature. In Brassard [6], pages 218–238.

[20] S. Miyaguchi, S. Kurihara, K. Ohta, and H. Morita. 128-bit hash function (N-hash). *NTT Review*, (2):128–132, 1990.

[21] N. Modadugu, D. Boneh, and M. Kim. Generating RSA keys on a handheld using an untrusted server. In *RSA Conference 2000*, San Jose, CA, U.S.A., Jan. 2000.

[22] National Institute of Standards and Technology (NIST) (Computer Systems Laboratory). Secure hash standard. Federal Information Processing Standards Publication FIPS PUB 180-1, Apr. 1995.

[23] A. Perrig. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8)*, Philadelphia PA, USA, Nov. 2001.

[24] G. Poupard and J. Stern. Security analysis of a practical "on the fly" authentication and signature generation. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT '98*, number 1403 in Lecture Notes in Computer Science, pages 422–436. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1998.

[25] G. Poupard and J. Stern. On the fly signatures based on factoring. In Tsudik [36], pages 37–45.

[26] B. Preneel. *Analysis and design of cryptographic hash functions.* PhD thesis, Katholieke Universiteit Leuven (Belgium), 1993.

[27] B. Preneel, R. Govaerts, and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. In B. S. Kaliski, Jr., editor, *Advances in Cryptology – CRYPTO '97*, number 1294 in Lecture Notes in Computer Science. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1997.

[28] M. O. Rabin. Digitalized signatures. In R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.

[29] R. Rivest. The MD5 message-digest algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force, Apr. 1992.

[30] R. L. Rivest and A. Shamir. PayWord and MicroMint: Two simple micropayment schemes. In M. Lomas, editor, *Security Protocols—International Workshop*, volume 1189 of *Lecture Notes in Computer Science*, pages 69 – 88, Cambridge, United Kingdom, Apr. 1997. Springer-Verlag, Berlin Germany.

[31] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[32] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet. In Tsudik [36], pages 93–100.

[33] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[34] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In J. Kilian, editor, *Advances in Cryptology – CRYPTO '2001*, number 2139 in Lecture Notes in Computer Science, pages 355–367. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2001.

[35] D. Tennenhouse. Embedding the Internet: Proactive computing. *Communications of the ACM*, 43(5):43–43, 2000.

[36] G. Tsudik, editor. *5th ACM Conference on Computer and Communications Security*, Singapore, Nov. 1999. ACM Press.

[37] B. Vöcking. How asymmetry helps load balancing. In *IEEE Symposium on Foundations of Computer Science*, pages 131–141, 1999.

[38] B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister. Smart dust: Communicating with a cubic-millimeter computer. *IEEE Computer*, pages 44–51, Jan. 2001.

[39] R. Yuster and U. Zwick. Finding even cycles even faster. In *Proceedings of the 21st International Colloquium on Automata, Languages and Programming*, volume 21, pages 532–543. Springer-Verlag, Berlin Germany, 1994.